AP ZW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Darpan Dinker
Mahesh Kannan
Pramod Gopinath

Serial No. 10/074,092

Filed: February 12, 2002

For: Distributed Data System with Process Co-Location and Out-of-Process Communication

§ Group Art Unit: 2155
§
§ Examiner: Duong, Oanh L.
§
§ Atty. Dkt. No.: 5681-05700
§ P7010

CERTIFICATE OF TRANSMISSION
37 C.F.R. § 1.8

I hereby certify that this correspondence is being facsimile transmitted to the U.S. Patent and Trademark Office (Fax No. (571) 273-8300), on the date indicated below:

Robert C. Kowert
Name of Registered Representative

September 1, 2006
Date

Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Panel Decision from Pre-Appeal Brief Review mailed August 1, 2006, Appellants present this Appeal Brief. **No extension of time should be due since this Appeal Brief is submitted by the deadline set by the Notice of Panel Decision.** Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

I. REAL PARTY IN INTEREST

The subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences at the time of the filing of this Brief.

III. STATUS OF CLAIMS

Claims 1 – 33 stand finally rejected. The rejection of claims 1 – 33 is being appealed. A copy of claims 1 – 33 as currently pending is included in the Claims Appendix herein below.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a distributed system including a plurality of nodes coupled together. Each of the plurality of nodes is coupled to at least one other of the plurality of nodes for communicating data between the nodes. *See, e.g.*, FIG. 3, 350A – D; page 4, lines 3 – 8; page 10, lines 3 – 20. The plurality of nodes includes at least one in-process node comprising an in-process client, a distributed data manager that executes on the same computer process on the in-process node as the in-process client, and at least one out-of-process node comprising an out-of-process client. *See, e.g.*, FIG. 3, 350A – D, 306A – D and 304A-D; page 11 – 1 – 10. Thus, the system of claim 1 provides for the simultaneous use of different node configurations for distributed data management within the same distributed system and in which different nodes may operate using different data management configurations. For example, a distributed data manager in a cluster may share its process space with a client while another distributed data manager may be out-of-process from its clients. *See, e.g.*, FIG. 3; page 4, lines 3 – 8; page 6, lines 4 – 14; page 10, lines 3 – 20; page 11 – 1 – 10.

The distributed data manager for the in-process node is configured to communicate data with the in-process client in a non-serialized format and communicate data with other ones of the plurality of nodes in a serialized format. *See, e.g.*, FIG. 3 - 5; page 4, lines 3 – 8; page 6, lines 4 – 14; page 11, lines 12 – 24; page 12, lines 12 – 23. The out-of-process client is configured to execute within a different process than any distributed data manager and is also configured to communicate data with other processes or other ones of the nodes in a serialized format. Since distributed data communication between nodes and for out-of-process nodes cross a process boundary, the object being communicated is serialized by the sending process and then send in a serialized format to the requesting node or client. *See, e.g.*, FIG. 3 - 6; page 4, lines 3 – 8; page 6, lines 4 – 14 and lines 28 - 30; page 11, lines 12 – 25;

Independent claim 17 is directed to a method including an in-process client requesting data from a distributed data manager for an in-process node of a distributed

data system. The in-process client and the distributed data manager for the in-process node execute within the same process on the in-process node. Thus, a client may request data within its local node. *See, e.g.*, FIG. 3 - 6, page 6, lines 16 – 30; page 10, lines 22-30; page 12, lines 12 – 23. The method of claim 17 also includes, if the requested data is present in a data store managed by the distributed data manager for the in-process node, the distributed data manager for the in-process node returning the requested data to the in-process client as an object without serializing the data. *See, e.g.*, FIG. 3 - 6, page 6, lines 16 – 30; page 10, lines 22- 30; page 12, lines 12 – 23.

If the requested data is not present in the data store managed by the distributed data manager for the in-process node, the method includes the distributed data manager for the in-process node retrieving the requested data in a serialized format from another node of the distributed data system, de-serializing the data retrieved from another node into an object, and returning the requested data to the in-process client as the de-serialized object. *See, e.g.*, FIG. 3 - 6, page 6, lines 16 – 30; page 12, lines 12 – 23.

Additionally, the method of claim 17 includes an out-of-process client requesting data from a node within the distributed data system and the out-of-process client receiving the requested data in a serialized format. An out-of-process client may request data and it may be requested from another node if the requested data is not locally present. The requested data will be received in a serialized format from another node having the data. *See, e.g.*, FIG. 3 - 6, page 7, lines 1 – 15; page 13, lines 4 – 15; page 14, lines 4 – 15.

Independent claim 32 is directed to a method includes an out-of-process client requesting data from a distributed data manager for an out-of-process node of a distributed data system, where the out-of-process client and the distributed data manager for the out-of-process node execute in two distinct processes. *See, e.g.*, FIG. 3 - 6, page 7, lines 1 – 15; page 13, lines 4 – 15. The method of claim 32 also includes that if the requested data is present in a data store managed by the distributed data manager for the out-of-process node, the distributed data manager for the out-of-process node returning

the requested data to the out-of-process client as a serialized object. *See, e.g.,* FIG. 3 – 7, page 6, lines 3- 8; page 7, lines 1 – 15; page 13, lines 4 – 15; page 14, lines 4 – 15.

The method also includes, if the requested data is not present in the data store managed by the distributed data manager for the out-of-process node, the distributed data manager for the out-of-process node retrieving the requested data in a serialized format from another node of the distributed data system and returning the requested data in a serialized format to the out-of-process client. An out-of-process client may request data and it may be requested from another node if the requested data is not locally present. The requested data will be received in a serialized format from another node having the data. *See, e.g.,* FIG. 3 – 7, page 6, lines 3- 8; page 7, lines 1 – 15; page 13, lines 4 – 15; page 14, lines 4 – 15.

The method of claim 32 includes an in-process client requesting data from a distributed data manager for an in-process node of the distributed data system, where the in-process client and the distributed data manager for the in-process node execute within the same process on the in-process node. *See, e.g.*, FIG. 3 - 6, page 6, lines 16 – 30; page 10, lines 22- 30; page 12, lines 12 – 23. Additionally, the method includes the in-process client receiving the requested data in de-serialized format. If the requested data is present in a data store managed by the distributed data manager for the in-process node, the distributed data manager may return the requested data to the in-process client without serializing the data. If the requested data is not present in the data store managed by the distributed data manager for the in-process node, the distributed store may retrieve the requested data in a serialized format, deserialize it and return the requested data to the in-process client. *See, e.g.*, FIG. 3 - 6, page 6, lines 16 – 30; page 10, lines 22- 30; page 12, lines 12 – 23.

The summary above describes various examples and embodiments of the claimed subject matter; however, the claims are not necessarily limited to any of these examples and embodiments. The claims should be interpreted based on the wording of the respective claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-9, 13-24 and 28-33 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over AAPA in view of Jin et al. (U.S. Patent 6,330,689) (hereinafter "Jin").

2. Claims 10-12 and 25-27 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over AAPA in view of Jin and further in view of Crites et al. (U.S. Patent 6,097,380) (hereinafter "Crites").

VII. ARGUMENT

First Ground of Rejection:

Claims 1-9, 13-24 and 28-33 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over AAPA in view of Jin et al. (U.S. Patent 6,330,689) (hereinafter "Jin"). Appellants traverse this rejection for at least the following reasons. Separately argued claims or claim groups are identified in the subheadings below.

Claims 1 - 5, 7-9, 13 – 20, 21-24, and 28-33:

The rejection is improper because the art relied upon by the Examiner explicitly teaches away from a distributed data system comprising a plurality of nodes including at least one in-process node and at least one out-of-process node, as recited in claim 1.

In rejecting claim 1, the Examiner acknowledges that AAPA (the background section of the present application) "does not explicitly teach wherein the plurality of nodes in the system comprises at least one in-process node and at least one [out-of-]process node". In fact, as pointed out in Appellants' previous responses, the background section clearly and unequivocally teaches: "Conventional systems allow ***only*** **one type of configuration** - either every node is an in-process node or every node is an out-of-process node. For example, if an out-of-process client is desired, then all other clients would also need to be configured as out-of-process clients" (Page 4, lines 20 – 23). See also, with respect to Figure 1: "This configuration is referred to as out-of-process and **all** nodes in the distributed data system in this configuration are configured out-of-process"(Page 1, line 31 – page 2, line 3) and with respect to Figure 2: "This configuration is called in-process and **all** nodes in the distributed data system in this configuration are configured in-process" (Page 3, lines 16 – 18). Thus, the AAPA reference relied upon by the Examiner in rejecting claim 1 **explicitly teaches away** from the combination of limitations recited in claim 1. "References that teach away cannot

serve to create a prima facie case of obviousness." *In re Gurley*, 27 F.3d 551, 553, 31 USPQ2d 1131, 1132 (Fed. Cir. 1994). Appellants note that on page 14 of the Final Action and in the Advisory Action the Examiner responds to the above argument by asserting, "a prior art reference must be considered in its entirety, including portions that would lead away from the claimed invention". However, when the AAPA reference **is** taken in its entirety, it clearly and unequivocally teaches away from the combination of limitations recited in claim 1, and therefore **cannot create a *prima facie* case of obviousness**.

Further with respect to claim 1, the Examiner asserts that "Jin teaches a server architecture wherein application can be run either in-process or out-of-process with the server program (see abstract)", and that "Jin teaches a data system comprising wherein the plurality of nodes in the system comprises at least one in-process node and at least one process node (col. 6 lines 42 – 48)." The Examiner then further asserts, "it would have been obvious to one of ordinary skill in the art to modify AAPA to include both in-process and out-of-process nodes as in Jin" and that "[o]ne would be motivated to do so to offer the flexibility to run either or both in-process and out-of-process applications (Jin, col. 6 lines 56 – 57)." In the Advisory Action, the Examiner asserts, "Jin teaches in-process node of a distributed data system" and "AAPA further teaches the in-process node comprises a client and a distributed data manager configured to execute within the same process", citing column 6, lines 42-57 of Jin and FIG. 2 as well as page 4, lines 12-18 of the AAPA. Appellants strongly traverse the Examiner's assertions for a number of reasons.

First, as pointed out in Appellant's previous responses, contrary to the Examiner's suggestion, Jin does not teach or suggest an in-process node of a distributed data system, where the in-process node comprises a client and a distributed data manager configured to execute within the same process, as recited in claim 1. In contrast, Jin teaches "application managers" that may run "within a server's process (i.e., in-process)" or "within separate processes (i.e., out-of-process)".

Moreover, as pointed out above, since the **AAPA reference teaches away** from the claimed combination of limitations in claim 1, the AAPA reference cannot be relied upon to create a *prima facie* case of obviousness. The Examiner asserts on page 15 of the Final Action, that "one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references". However, Appellants only discuss the references individually to show that the Examiner's reliance on the individual references is misplaced. It is clear that the references, whether considered alone or in combination, do not teach or suggest a distributed data system comprising a plurality of nodes including at least one in-process node and at least one out-of-process node, as recited in claim 1. Since the AAPA reference explicitly teaches away from the claimed combination of limitations, and since the Examiner agrees that Jin does not teach the claimed limitations under discussion, the art cited by the Examiner, when considered in its totality, fails to render the claimed combination of limitations obvious.

Further with respect to claim 1, Appellant pointed out in the previous response that there is no teaching or suggestion in the art of record that Jin's in-process and out-of-process "application manager" and "server" combinations would in any way apply to the configuration of clients and distributed data managers in the nodes of AAPA. The Examiner does not appear to address this argument in the Final Action. Appellants submit that Jin's use of the terms "in-process" and "out-of-process" refers to the manner in which Jin's "application managers" responsible for processing client requests interact with Jin's "server processes". **Jin does not teach or suggest a client executing within the same process as a distributed data manager, much less an in-process node of a distributed data system at which a distributed data manager and a client execute within the same process, as recited in claim 1.**

Still further with respect to claim 1, as also pointed out in Appellants' previous responses, even if the teachings of Jin were to be combined with those of AAPA, this would only lead to "application managers" (client request processors) that may "run within the same process" or "different processes" as a server, not to a distributed data system in which at some nodes, clients run in the same process as distributed data

managers, and at other nodes, clients run in separate processes from distributed data managers, as recited in claim 1. Appellants note that on page 16 of the Final Action, the Examiner suggests that one would be motivated to combine the teachings of AAPA and Jin because this would "offer developers the flexibility to run either or both in-process and out-of-process applications (Jin, col. 6 lines 55-57), thereby providing higher performance at the risk of crashing the system and affording higher reliability (Jin, col. 4, lines 55-62). The Examiner is incorrect. AAPA expressly teaches away from the combination proposed by the Examiner. As noted above, AAPA teaches that in conventional distributed data systems, all nodes must be either in-process or out-of-process (no mixing of in-process and out-of-process nodes is allowed). It is improper to combine references where the references teach away from their combination. *In re Grasselli*, 218 USPQ 769, 779 (Fed. Cir. 1983). Therefore, the rejection is clearly improper.

Claims 6 and 21:

In regards to claim 6 and contrary to the Examiner's assertion, the combination of AAPA and Jin fails to teach or suggest wherein **all** data store operations performed by the distributed data manager in the in-process node store data in a non-serialized object format in a data store of the in-process node. The Examiner cites page 4, lines 15 – 18 and lines 8 – 10 of the AAPA.

However, **the Examiner is incorrect.** The AAPA at page 4, lines 15 – 18 teaches that "in-process configuration data may be communicated between a distributed data manager and a client sharing the same process space, without the additional computation requirement for serialization/deserialization" and lines 8 – 10 teach that "a requesting distributed data manager stores data in its data store and returns the data or a pointer to the client 201A indicating where the requested data is in the data store 221A", but nowhere in the cited art is there a teaching or suggestion of **all** data store operations performed by a distributed data manager in an in-process node storing data in a non-serialized format in a data store of the in-process node, as recited in the claim.

Appellant notes that on pages 16 – 17 of the Final Action, the Examiner suggests that "one of ordinary skill in the art will readily recognize that data is stored in a non-serialized format in a data store since data is returned to client from data store without serialization/deserialization". Appellants submit that the Examiner's reasoning is incorrect. Just because a distributed data manager and a client may communicate in-process configuration data without serialization/deserialization and may store data in its data store does not thereby render obvious the limitation that **all** data store operations performed by the distributed data manager in the in-process node store data *in a non-serialized object format in a data store.* The rejection of claim 6 is therefore further unsupported by the cited art, and removal thereof is respectfully requested. Similar remarks also apply to claim 21.

Second Ground of Rejection:

Claims 10-12 and 25-27 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over AAPA in view of Jin and further in view of Crites et al. (U.S. Patent 6,097,380) (hereinafter "Crites"). Appellants traverse this rejection for at least the following reasons. Separately argued claims or claim groups are identified in the subheadings below.

Claims 10 – 12 and 25 - 27:

With respect to claim 10, the Examiner acknowledges that AAPA-Jin does not teach "the distributed data manager for the out-of-process node is configured to store the data in its serialized format". The Examiner relies on Crites, citing col. 2, lines 36 – 42 and col. 5, lines 19 – 20 of Crites. However, the cited portion of Crites merely states that "the continuous media streams consist of sequences of digital data that are intended to be supplied serially to client computers and rendered by the client computers in a form that is useful to users of the client computers". Crites provides an example of a continuous media stream as "a sequence of audio samples" that are sent to a "client computer" for

"rendering as audible sound". In contrast, claim 10 recites the "out-of-process client" sending "serialized data" to the distributed data manager at an out-of-process node of a distributed data system, and the distributed data manager then storing the data in serialized format. In Crites, the "media stream" is sent to the client, whereas in claim 10, serialized data is sent from the client to a distributed data manager for storage.

In response, the Examiner asserts on page 17 of the Final Action:

> "Crites teaches storing data in serialized format (Examiner has given a broadest reasonable interpretation of "data in serialized format" as "data stream" in view of specification (pages 3 lines 7 – 8). Crites teaches mass storage devices are stored a plurality of data streams, col. 2 lines 36 – 38. Therefore, storing data stream, disclosed by Crites, reads on storing data in serialized format ...)"

The Examiner's interpretation is incorrect. The cited portion of AAPA teaches that "serialization may include generating object data sequentially so that it may be transmitted as a data stream", but this does not imply that storing a "continuous media data stream", as taught in Crites, somehow represents data being stored in a serialized format.

Additionally, in Crites, the "media stream" is sent to the client, whereas in claim 10, serialized data is sent from the client to a distributed data manager for storage. In response, the Examiner cites portions of AAPA that also fail to teach the claimed limitation wherein the out-of-process client is configured to send serialized data to the distributed data manager for the out-of-process node to store data.

As pointed out in Appellant's previous responses, even if the teachings of Crites were somehow to be combined with those of Jin and AAPA, the resulting system would merely allow media streams to be served from the nodes of a homogeneous distributed data system (i.e., one in which the nodes are either all "in-process nodes" or all "out-of-process nodes"), and would not render the combination of limitations of claim 10 obvious. Thus, the rejection of claim 10 is clearly unsupported by the cited art. Similar remarks also apply to claim 25.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-33 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-05700/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: September 1, 2006

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A distributed data system, comprising:

a plurality of nodes coupled together, wherein each one of the plurality of nodes is coupled to at least one other one of the plurality of nodes for communicating data between the nodes, wherein the plurality of nodes comprises:

at least one in-process node comprising:

an in-process client; and

a distributed data manager, wherein the in-process client and the distributed data manager for the in-process node are configured to execute within the same computer process on the in-process node, and wherein the distributed data manager for the in-process node is configured to communicate data with the in-process client in a non-serialized format and communicate data with other ones of the plurality of nodes in a serialized format; and

at least one out-of-process node comprising an out-of-process client, wherein the out-of-process client is configured to execute within a different process than any distributed data manager, and wherein the out-of-process client is configured to communicate data with other processes or other ones of the nodes in a serialized format.

2. The system as recited in claim 1, wherein the in-process client is configured to request data from the distributed data manager for the in-process node, and

wherein, in response to the client's request, the distributed data manager for the in-process node is configured to return a reference to an object for the data without serializing the data.

3. The system as recited in claim 2, wherein the distributed data manager for the in-process node is configured to receive a request for data from another node, serialize the requested data and send the serialized data to the requesting node.

4. The system as recited in claim 1, wherein the distributed data manager for the in-process node is configured to receive serialized data from another node, de-serialize the data and store the de-serialized data as an object.

5. The system as recited in claim 1, wherein said in-process client is configured to send an object reference for the data to the distributed data manager for the in-process node to store data in the in-process node.

6. The system as recited in claim 1, wherein all data store operations performed by the distributed data manager in the in-process node store data in a non-serialized object format in a data store of the in-process node.

7. The system as recited in claim 1, further comprising a distributed data manager for the out-of-process node, wherein the distributed data manager for the out-of-process node is configured to communicate data with other processes or other ones of the plurality of nodes in a serialized format.

8. The system as recited in claim 7, wherein the out-of-process client is configured to execute in a separate process and communicate data with the distributed data manager for the out-of-process node in a serialized format.

9. The system as recited in claim 7, wherein the out-of-process client is configured to request data from the distributed data manager for the out-of-process node,

and wherein, in response to the client's request, the distributed data manager for the out-of-process node is configured to send the requested data to the out-of-process client in a serialized format.

10. The system as recited in claim 7, wherein said out-of-process client is configured to send serialized data to the distributed data manager for the out-of-process node to store data, wherein the distributed data manager for the out-of-process node is configured to store the data in its serialized format.

11. The system as recited in claim 7, wherein the distributed data manager for the out-of-process node is configured to receive serialized data from another node, and store the data in its serialized format.

12. The system as recited in claim 7, wherein all data store operations performed by the distributed data manager for the out-of-process node store data in a serialized format in a data store of the out-of-process node.

13. The system as recited in claim 7, wherein the distributed data manager for the out-of-process node is configured to replicate data to one or more other ones of the plurality of nodes.

14. The system as recited in claim 7, wherein the distributed data manager for the out-of-process node is comprised within an application server, and wherein the out-of-process client is a web server coupled to the application server.

15. The system as recited in claim 1, wherein the distributed data manager for the in-process node is configured to replicate data stored in the in-process node to one or more other ones of the plurality of nodes.

16. The system as recited in claim 1, wherein the distributed data manager for the in-process node is comprised within an application server, and wherein the in-process

client is a web server coupled to the application server.

17. A method, comprising:

an in-process client requesting data from a distributed data manager for an in-process node of a distributed data system, wherein the in-process client and the distributed data manager for the in-process node execute within the same process on the in-process node;

if the requested data is present in a data store managed by the distributed data manager for the in-process node:

the distributed data manager for the in-process node returning the requested data to the in-process client as an object without serializing the data;

if the requested data is not present in the data store managed by the distributed data manager for the in-process node:

the distributed data manager for the in-process node retrieving the requested data in a serialized format from another node of the distributed data system;

the distributed data manager for the in-process node de-serializing the data retrieved from another node into an object; and

the distributed data manager for the in-process node returning the requested data to the in-process client as the de-serialized object;

an out-of-process client requesting data from a node within the distributed data system; and

the out-of-process client receiving the requested data in a serialized format.

18. The method as recited in claim 17, further comprising:

the distributed data manager for the in-process node receiving a request for data from another node;

the distributed data manager for the in-process node serializing the requested data; and

the distributed data manager for the in-process node sending the serialized data to the requesting node.

19. The method as recited in claim 17, further comprising the distributed data manager for the in-process node receiving serialized data from another node, de-serializing the data and storing the de-serialized data as an object.

20. The method as recited in claim 17, further comprising said in-process client sending an object reference for the data to the distributed data manager for the in-process node for storing data in the in-process node.

21. The method as recited in claim 17, wherein all data store operations performed by the distributed data manager in the in-process node comprise storing data in a non-serialized object format in a data store of the in-process node.

22. The method as recited in claim 17, wherein the node within the distributed data system from which the out-of-process client requests data is configured as an out-of-process node comprising a distributed data manager, wherein the distributed data

manager for the out-of-process node is configured to communicate data with other processes or other nodes of the distributed data system in a serialized format.

23. The method as recited in claim 22, further comprising the out-of-process client executing in a separate process and communicating data with the distributed data manager for the out-of-process node in a serialized format.

24. The method as recited in claim 22, further comprising:

the out-of-process client requesting data from the distributed data manager for the out-of-process node; and

the distributed data manager for the out-of-process node sending the requested data to the out-of-process client in a serialized format in response to the client's request.

25. The system as recited in claim 22, further comprising said out-of-process client storing data in the out-of-process node by sending serialized data to the distributed data manager for the out-of-process node, wherein the distributed data manager for the out-of-process node is configured to store the data in its serialized format.

26. The method as recited in claim 22, further comprising the distributed data manager for the out-of-process node receiving serialized data from another node, and storing the data in its serialized format.

27. The method as recited in claim 22, wherein all data store operations performed by the distributed data manager for the out-of-process node comprise storing data in a serialized format in a data store of the out-of-process node.

28. The method as recited in claim 22, further comprising the distributed data manager for the out-of-process node replicating data stored in the out-of-process node to

other nodes of the distributed data system in a serialized format.

29. The method as recited in claim 22, wherein the distributed data manager for the out-of-process node is comprised within an application server, and wherein the out-of-process client is a web server coupled to the application server.

30. The method as recited in claim 17, further comprising the distributed data manager for the in-process node replicating data stored in the in-process node to other nodes of the distributed data system in a serialized format.

31. The method as recited in claim 17, wherein the distributed data manager for the in-process node is comprised within an application server, and wherein the in-process client is a web server coupled to the application server.

32. A method, comprising:

an out-of-process client requesting data from a distributed data manager for an out-of-process node of a distributed data system, wherein the out-of-process client and the distributed data manager for the out-of-process node execute in two distinct processes;

if the requested data is present in a data store managed by the distributed data manager for the out-of-process node:

the distributed data manager for the out-of-process node returning the requested data to the out-of-process client as a serialized object;

if the requested data is not present in the data store managed by the distributed data manager for the out-of-process node:

the distributed data manager for the out-of-process node retrieving the requested data in a serialized format from another node of the distributed data system; and

the distributed data manager for the out-of-process node returning the requested data in a serialized format to the out-of-process client;

an in-process client requesting data from a distributed data manager for an in-process node of the distributed data system, wherein the in-process client and the distributed data manager for the in-process node execute within the same process on the in-process node; and

the in-process client receiving the requested data in de-serialized format.

33. The method as recited in claim 32, further comprising the distributed data manager for the out-of-process node storing the data retrieved from another node in a serialized format in the data store managed by the distributed data manager for the out-of-process node.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.